

**WHAT IS CLAIMED IS:**

*Sub*  
*at*  
1. A method of debugging code that executes in a multithreaded processor having a plurality of microengines comprises:

inserting a segment of executable code into an unused section of a target microengine's microstore in response to a first context swap of one of a plurality of hardware-supported execution threads of a program executing in the target microengine;

executing the segment of executable code; and  
resuming execution of the program in response to a second context swap.

2. The method of claim 1 wherein inserting comprises:  
saving program counters associated with the plurality of hardware-supported execution threads;

modifying the target microengine's program counters to jump to a start of the segment of executable code; and

appending the saved program counters to an end of the segment of executable code.

3. The method of claim 1 wherein inserting further comprises receiving a user request to execute the segment of executable code.

1 4. The method of claim 1 wherein executing further comprises  
2 examining states of the execution threads at the second  
3 context swap.

1 5. The method of claim 1 wherein resuming further comprises  
2 removing the segment of executable code from the microstore.

1 6. The method of claim 1 wherein resuming comprises  
2 restoring the programs counters of the plurality of hardware-  
3 supported execution threads that have not executed.

1 7. The method of claim 1 wherein the segment of executable  
2 code resides in a library executable code segments residing in  
3 the processor.

1 8. A method of debugging software that executes in a  
2 multithreaded processor having a plurality of microengines  
3 comprises:

4 pausing program execution in a plurality of threads of  
5 execution within a target microengine;

6 inserting a segment of executable code into an unused  
7 section of the target microengine's microstore;

8 executing the segment of executable code in the target  
9 microengine; and

10 resuming program execution in the target microengine.

1 9. The method of claim 8 wherein pausing is in response to a  
2 user command to pause.

1 10. The instruction of claim 8 wherein the user command to  
2 pause further comprises selecting the target microengine from  
3 one of the plurality of microengines.

1 11. The method of claim 10 wherein the user command to pause  
2 further comprises selecting the segment of executable code.

1 12. The method of claim 8 wherein pausing further comprises  
2 determining when one of the plurality of threads of execution  
3 context swaps.

1 13. The method of claim 8 wherein inserting further  
2 comprises:

3 modifying a program counter of the paused program to  
4 point to the segment of executable code; and  
5 modifying a program counter at an end of the segment of  
6 executable code to point to the paused program.

1 14. The method of claim 8 wherein the segment of executable  
2 code causes the target microengine to write to specific  
3 registers.

1 15. The method of claim 14 wherein the specific registers are  
2 examined by a user during execution of the segment of  
3 executable code.

1 16. A processor that can execute multiple contexts and that  
2 comprises:

3 a register stack;

4 a program counter for each executing context;  
5 an arithmetic logic unit coupled to the register stack  
6 and a program control store that stores a breakpoint command  
7 that causes the processor to:  
8 pause program execution in a context in the processor;  
9 insert a segment of executable code into an used section  
10 of a microstore associated with the context;  
11 execute the segment of executable code; and  
12 resume program execution.

1 17. The processor of claim 16 wherein program execution is  
2 paused by disabling a processor enable bit.

1 18. The processor of claim 16 wherein the segment is inserted  
2 in response to a user request received through a remote user  
3 interface connected to the processor.

1 19. The processor of claim 16 wherein an end of the segment  
2 points to a program counter of the program.

1 20. The processor of claim 16 wherein the segment examines  
2 states of execution of the contexts.

1 21. The processor of claim 16 wherein program execution is  
2 resumed by enabling a processor enable bit.

1 22. A computer program product, disposed on a computer  
2 readable medium, the product including instructions for  
3 causing a multithreaded processor having a plurality of  
4 microengines to:

5 pause program execution in a plurality of threads of  
6 execution within a target microengine;

7 insert a segment of executable code into an unused  
8 section of the target microengine's microstore;

9 execute the segment of executable code in the target  
10 microengine; and

11 resume program execution in the target microengine.